Received at NIC 18-June-74

30239

Network Working Group
Request for Comments: #632
NIC #30239

H. Opderbeck
UCLA - NMC
Computer Science
20 MAY 74

# Throughput Degradations for Single Packet Messages

The transmission of digitized speech over the ARPANET represents a new dimension in the use of packet switching systems. The throughput and delay requirements for this newly emerging application area are quite different from the throughput and delay requirements for interactive use or file transfers. In particular, we need to achieve a high throughput for small messages since long messages result in long source delays to fill the large buffers. Therefore we are currently studying the throughput limits for single-packet messages. We realize that up to now little attempt was made to optimize throughput for low delay traffic. It was nevertheless surprising for us to find out that the observed throughput for single-packet messages is in many cases only about one fourth of what one would expect. In what follows we are going to explain why this happens and what could be done to correct this situation.

On April 1, 1974, we sent, using the IMP message generator, single-packet messages at the highest possible rate ("RFNM-driven") from the MOFFET-IMP to the SRI-IMP. There are two three-hop paths from MOFFET to SRI, one of them involving two 230.4 kbs circuits. Since there was hardly any interfering traffic we expected an average round-trip delay of not more than 100 msec. Assuming that there are, on an average, 3 messages in transmission between MOFFET and SRI and assuming a message length of about 1000 bits this should result in a throughput of more than 30 kbs. The observed throughput was, however, less than 8 kbs. A repetition of the experiment showed the same result. A more detailed analysis of the collected data revealed that an average number of 3.5 messages were simultaneously in transmission between MOFFET and SRI. The throughput degradation could therefore not have been due to interfering traffic between these two sites. Also the channel utilization for all channels that were involved in the transmission was less than 40 percent. The observed mean round-trip times between MOFFET and SRI, however, were about 500 msec. Since these large round-trip times were obviously not due to physical limitations, we studied the flow control mechanism for single-packet messages and were able to come up with an explanation for this undesirable behavior.

When a single-packet message arrives at the destination IMP out of order (i.e., the logically preceding message has not yet arrived

there) it is not accepted by the destination IMP. It is rather treated as a request for the allocation of one reassembly buffer. The corresponding ALLOCATE is then sent back to the source IMP only after the RFNM for the previous message has been processed. We therefore may have the following sequence of events:

1 MSG(i) sent from SOURCE-IMP (message i is sent from the source IMP to the destination IMP).

2 MSG(i + 1) sent from SOURCE-IMP.

3 MSG(i + 1)) arrives at DEST-IMP (due to an alternate path or a line error, message i + 1) aarrives at the destination IMP out of order; it is treated as a request for one reassembly buffer allocation and then discarded).

4 MSG(i) arrives at DEST-IMP (messages i arrives at the destination IMP; it is put on the proper HOST output queue).

5 RFNM(i) sent from DEST-IMP (after message i has been accepted by the destination HOST the RFNM is sent to the source IMP).

6 ALL(i + 1) sent from DEST-IMP (only after the RFNM for message i has been processed can the ALLOCATE for message i + 1 be sent).

7 RFNM(i) arrives at SOURCE-IMP.

8 ALL(i + 1) arrives at SOURCE-IMP.

9 MSG(i + 1) arrives at DEST-IMP (now message i + 1 is put on the proper HOST output queue).

10 RFNM(i + 1) sent from DEST-IMP.

11 RFNM(i + 1) arrives at SOURCE-IMP.

Note that the round-trip time for message i + 1 is the time interval between event 2 and event 12. Therefore the round-trip time for message i + 1 is more than twice as large as it would have been if it had arrived in order, other conditions being unchanged. Therefore a line error will in many cases not only delay the message in error but also the next single-packet message if this message follows the preceding message within 125 msec, the error retransmission timeout interval. Also, a faster, alternate path to the destination IMP can actually slow down the transmission since it causes messages to arrive there out of order.

This situation becomes even worse when we consider RFNM-driven single-packet message traffic. Table 1 shows a possible sequence of events. We again assume that message i + 1 reaches the destination IMP before message i.

| SOURCE IMP | DESTINATION IMP |
|---|---|
| MSG(i) sent | |
| MSG(i + 1) sent | |
| MSG(i + 2) sent | MSG(i + 1) arr |
| MSG(i + 3) sent | MSG(i) arr |
| | RFNM(i) sent |
| | ALL(i + 1) sent |
| | MSG(i + 2) arr |
| | MSG(i + 3) arr |
| RFNM(i) arr | |
| MSG(i + 4) sent | |
| ALL(i + 1) arr | MSG(i + 4) arr |
| MSG(i + 1) sent | MSG(i + 1) arr |
| | RFNM(i + 1) sent |
| RFNM(i + 1) arr | ALL(i + 2) sent |
| MSG(i + 5) sent | |
| ALL(i + 2) arr | MSG(1 + 5) arr |
| MSG(i + 2) sent | MSG(i + 2) arr |
| | RFNM(1 + 2) sent |
| RFNM(i + 2) arr | ALL(i + 3) sent |
| MSG(i + 6) sent | |
| ALL(i + 3) arr | MSG(i + 6) arr |
| MSG(i + 3) sent | MSG(i + 3) arr |
| | RFNM(i + 3) sent |
| RFNM(i + 3) arr | ALL(i + 4) sent |
| MSG(i + 7) sent | |
| ALL (i + 4) arr | MSG(i + 7) arr |
| MSG(i + 4) sent | MSG(i + 4) arr |
| | RFNM(i + 4) sent |
| RFNM(i + 4) arr | ALL(i + 5) sent |
| MSG(i + 8) sent | |
| ALL(i + 5) arr | |
| MSG(i + 5) sent | |

Table 1.

Retransmission Pattern for the Current Flow Control Scheme

(Since the traffic is RFNM-driven, the arrival of RFNM i, i + 1,... is followed by the sending of messages i + 4, i + 5,,...)

The most interesting fact about this sequence of events is that the arrival of message i + 1 before message i at the destination IMP causes not only message i + 1 but all future messages to be retransmitted--though we do not assume that any of the future messages arrive out of order. The table also shows that the

round-trip times for message i + 4 and all future messages is more than four times as large as it would be without these undesirable retransmissions. it is also noteworthy that, once this retransmission pattern has established itself, there is almost no way the system can recover from this condition other than interrupting the input stream at the source IMP. A single arrival out of order of any of the later user or control messages, for instance, will not change this retransmission pattern. The normal flow of single-packet messages will only reestablish itself if, for example, messages i + 4, i + 5, and i + 6 are simultaneously delayed for several hundred milliseconds such that messages i + 1, i + 2, and I + 3 can be retransmitted in the meantime. The probability of occurrence of such an event is, however, extremely small. Therefore one can consider the system as being trapped in this undesirable retransmission condition. The "normal" flow of messages, on the other hand, represents only the transient behavior of the system since there is always a finite probability that two messages arrive out of order due to transmission errors.

As mentioned before, the system can only recover from this throughput (and delay) degradation if the input stream of single-packet messages is interrupted. In case of speech transmission, however, this might not occur for a long time. Therefore speech transmission systems would in many cases have to work with only one fourth of the expected single-packet bandwidth. Since this is clearly an unacceptable condition we looked for a modification of the current flow control scheme which corrects this situation. In what follows we describe two methods that could be used to avoid the undesirable retransmission of messages.

Recall that a single-packet message is rejected at the destination IMP and later retransmitted if the RFNM for the preceding message has not yet been sent to the source IMP. This is mainly done to prevent the occurrence of reassembly lockup conditions [1]. Therefore the problem cannot be solved by simply accepting all single-packet messages without additional measures to prevent deadlocks. This could lead to a reassembly lockup if a large number of single-packet messages from several source IMPs arrives at their common destination IMP out of order. In this case the destination IMP might not be able to accept those messages that are in order because of the lack of reassembly buffers. As a result the system is deadlocked. Any solution of the throughput degradation problem must guarantee that all messages that arrive in order can be accepted by the destination IMP.

One way to achieve this goal is to reject single-packet messages that arrive out of order only if the buffer requirement(s) of the preceding messages(s) is not known. In the previous example we have seen that the destination IMP continuously rejected messages although it knew the buffer requirements for the messages that had to be delivered first. As the buffer requirements become known, the

necessary number of buffers can be set aside and future single-packet messages can be accepted without the danger of deadlock. Therefore the undesirable retransmission pattern cannot establish itself. Table 2 shows the sequence of events for this policy if message i + 1 arrives before message i at the destination IMP.

| SOURCE IMP | DESTINATION IMP |
|---|---|
| MSG(i) sent | |
| MSG(i + 1) sent | |
| MSG(i + 2) sent | MSG(i + 1) arr. (rejected) |
| MSG(i + 3) sent | MSG(i) arr. (HOST output) |
| | RFNM(i) sent |
| | ALL(i + 1) sent |
| | MSG(i + 2) arr (stored) |
| | MSG(i + 3) arr (stored) |
| RFNM(i) arr | |
| MSG(i + 4) sent | |
| ALL(i + 1) arr | MSG(i +4) arr (stored) |
| MSG(i + 1) sent | MSG(i + 1) arr (HOST output) |
| | RFNM(i + 1) sent |
| RFNM(i + 1) arr | RFNM(i + 2) sent |
| MSG(i + 5) sent | RFNM(i + 3) sent |
| | RFNM(i + 4) sent |

Table 2.

Sequence of Events for Modified Flow Control Scheme

Note that in this modified scheme only one message, message i + 1, is retransmitted. In view of the fact that the IMPs have plenty of reassembly buffer space it is, however, desirable to avoid this one retransmission, too. This is particularly important for the transmission of speech which depends on a steady flow of data and will be disrupted by a sudden large delay. Therefore we suggest a second method to solve the throughput degradation problem which, in most cases, will not require any retransmissions.

Suppose all single-packet messages are initially accepted (or stored). Currently single-packet messages that arrive out of order are rejected because of the possibility of a deadlock. But let us take a closer look at the situation where all single-packet messages are accepted (or stored) such that there is no reassembly buffer available for messages that have to be delivered to their HOST(s) next. This is not really a lockup condition because the source IMPs keep a copy of all single-packet messages for which a RFNM has not yet been received. Therefore any single-packet message, which

arrived out of order but was accepted by the destination IMP nevertheless, can be deleted later without the message being lost. The destination IMP only has to send an ALLOCATE for each deleted single-packet message to the corresponding source IMP when reassembly buffer space is available. This can also be considered as a deferred rejection. But now a retransmission is only necessary if the destination IMP is really running out of reassembly buffers. In this case, the physical limitations of the system are reached and we cannot hope to gain large throughput increases by means of protocol changes.

It is our intention to pursue this issue with the IMP development group at BBN in the future. They agree that the two solutions we suggest would improve the situation. However, they can think of alternative solutions.

I acknowledge the help of Bill Naylor and Joe Katz in performing the experiments which led to the discovery of the throughput degradation.

References:

[1] Kleinrock, L. and H. Opderbeck. "On a Possible Lockup condition in the IMP Subnet Due to Message Sequencing", RFC #626.